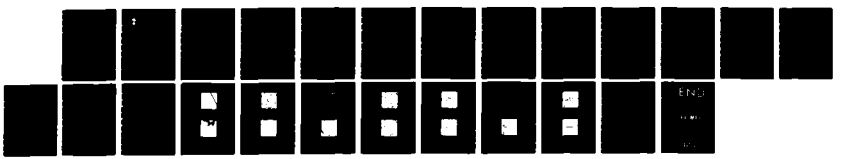
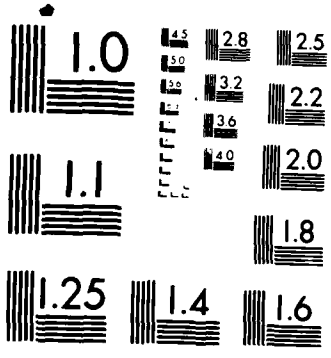
AD-A162 024 LINE FINDING ALGORITHMS FOR SAR (SYNTHETIC APERATURE RADAR)(U) ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) J W WOOD JUL 85 RSEE-MEMO-3841 DRIC-BR-97301 1/1

UNCLASSIFIED F/G 17/9 NL





UNLIMITED

RSRE
MEMORANDUM No. 3841

ROYAL SIGNALS & RADAR ESTABLISHMENT

LINE FINDING ALGORITHMS FOR SAR

Author: J W Wood

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

UNLIMITED

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3841

Title: LINE FINDING ALGORITHMS FOR SAR

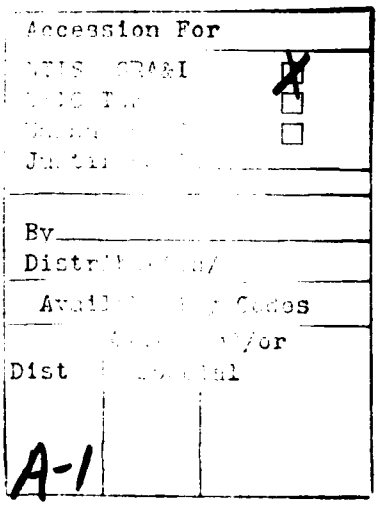
Author: J W Wood

Date: July 1985

SUMMARY

This memorandum addresses the problem of locating linear features, not necessarily straight, in Synthetic Aperture Radar images. The characteristics of such features and the way in which these characteristics affect the desired properties of line finding algorithms are discussed.

Two line detection algorithms are then described: the Hough transform, and the dynamic programming technique. Results of applying both algorithms to simulated and real SAR images are given. The Hough transform is found to be too restricted to be useful for this application. The dynamic programming technique locates lines well. The problem of spurious lines appearing in clutter is discussed.



RSRE MEMORANDUM NO 3841

LINE FINDING ALGORITHMS FOR SAR

J W Wood

CONTENTS

1 INTRODUCTION

Remotely sensed images of terrain contain a large number of different features: fields, trees, hedges, rivers, buildings, roads etc. The identification, extraction and analysis of interesting features from images can be a very time consuming task if done manually. If the image sensor produces data at a significant rate, this task may be prohibitively expensive. The RSRE X-band SAR is one example of such a system: images can be produced in real time over a swath 10 km wide, at an aircraft velocity of the order of 200 m/s. The resolution of the system is of the order of metres, and thus images are produced at the rate of approximately one million pixels per second. Interpretation of the images is complicated still further by the presence of speckle noise which is a consequence of the coherent nature of the imaging system.

Since human interpretation of the complete set of image data is not feasible, there is a requirement for an automatic system of image analysis. Even if this system is unable to perform a complete analysis of the image, it may still be used to select a subset of the image that can be presented to a higher level operator (either a human, or a more intelligent system that is too slow to be used for analysis of the complete image).

This memorandum describes some initial work that has been done towards defining one part of the low level image analysis system: namely that of detecting linear features in images. Examples of such features are roads, hedges and fences. Road detection is a worthwhile objective, since it is likely that other objects of interest may be found in their vicinity, particularly vehicles. Also, linear features can be used for image to image and image to map registration.

Section 2 discusses the effect of the speckle noise in SAR images on the appearance of lines in images, and the way this influences the required properties of a line finding algorithm. Sections 3 and 4 discuss two particular line detection algorithms in detail: the Hough transform and dynamic programming respectively. The results of applying the two algorithms to synthetic and real image data are presented in Section 5. Conclusions and suggestions for further work are contained in Section 6.

2 THE EFFECTS OF SPECKLE

SAR is a coherent imaging system. The major consequence of this fact is that the images are degraded by speckle noise. This noise is uncorrelated from resolution cell to resolution cell, and the probability density function of the intensity over a region of constant radar cross section is given by

$$p(I)dI = \frac{1}{\langle I \rangle} \exp\left(-\frac{I}{\langle I \rangle}\right) dI \qquad (2.1)$$

where $\langle I \rangle$ is the mean intensity over the region.

Suppose we have a region with mean intensity $\langle I_T \rangle$, against a background of mean intensity $\langle I_B \rangle$. Then we can set a threshold for detection of the region against the background at the point at which the two probability distributions are equal, given by

$$T = \left(\frac{\langle I_T \rangle \langle I_B \rangle}{\langle I_T \rangle - \langle I_B \rangle}\right) \ln\left(\frac{\langle I_T \rangle}{\langle I_B \rangle}\right) \qquad (2.2)$$

The probability that a single pixel in the region will be incorrectly classified as being part of the background is then given by

$$D = P(I_T < T) = 1 - \exp\left(-\frac{T}{\langle I_T \rangle}\right) \qquad (2.3)$$

For example, for a region whose mean intensity is 10 times that of the background, the probability that a single pixel in the region has an intensity less than the threshold is approximately 0.23. Fig 2.1 shows the behaviour of D as a function of $\langle I_T \rangle / \langle I_B \rangle$.

Speckle noise also affects the appearance of linear features in the image. Consider a line of mean intensity $\langle I_L \rangle$ against a background of mean intensity $\langle I_B \rangle$. Then we can define a threshold for detection of the line as above, and the probability of any single pixel on the line having an intensity below this threshold is given by 2.3 above. We also wish to calculate the probability P_R, of the gap of length R pixels in a line. Clearly this probability increases with the length, N, of the line, but it is not possible to find an exact formula for P_R.

In the absence of an analytic formula for P_R, a program was written to investigate the behaviour of P_R, as D, R and N were varied. This was done by generating sequences of up to N binary numbers with probability D of being set. If a sequence of R set numbers was detected, a success was recorded. 10000 trials were performed, and the probability of a gap of length R in a line of length N was thus given by the number of successes divided by 10000. 5 values of D were used, and the probabilities of gaps of length 1, 2, 3, 4 and 5 pixels in lines of lengths 5, 10, 15.....50 pixels were estimated. These results are tabulated in appendix 1, and Fig 2.2 shows the probabilities of gaps of various lengths for different length lines when the probability of a single pixel "dropping out" is 0.2, corresponding to a line which is ~12.22 times brighter than the background. Clearly there is a significant probability of a gap of length 3 pixels for lines which are 20 pixels or more in length.

These results shown that a local operator with a window size of less than seven pixels is unlikely to be successful in locating lines on SAR images. An algorithm is required that is capable of taking a more "global" view of the image, and spanning the gaps in the line due to speckle. Two such algorithms will be discussed in the following two sections.

3 THE HOUGH TRANSFORM

The Hough transform (eg Duda and Hart 1972) is a widely used line detection algorithm. It is a global procedure, in that all points that lie on a given line in an image map onto a single point in a transformed space.

3.1 IMPLEMENTATION

Any straight line can be parameterised by the angle that the normal to it makes with the horizontal, θ, and its distance r, from an origin, typically the centre of the image. (The "standard" parameterisation of a line, in terms of its slope and intercept with an axis is not appropriate here, since these quantities can increase without limit).

The Hough transform proceeds as follows: each point in the image lies on many possible straight lines, and thus maps onto many points in r, θ space. For an image point at x,y, the corresponding points in the transform space are given by

$$r = x \cos(\Theta) + y \sin(\Theta) \qquad (3.1)$$

i.e, a sinusoidal curve. The image intensity at (x,y) is added to the transform for each point on the curve. This procedure is repeated for all points in the image. If many points lie on the same straight line, their corresponding curves will intersect at a particular point, leading to a peak in the Hough transform. The location of these peaks gives the parameters (r,Θ) of the lines in the image.

The fact that all points in the image are used independently makes the method robust to missing points in the line - their effect is simply to reduce the height of the peak in the transform space. The discussion of section 2 showed that this is a desirable quality for a SAR image line finding algorithm.

When locating peaks, it is important to take into account the fact that, depending on the angle and distance of the line from the origin, different lengths of the line intersect the image, and so make different contributions to the Hough transform. For example, a line passing diagonally through the centre of the image clearly has a much longer intersection than one which passes through one corner. For this reason, the Hough transform must be normalised by dividing it by the transform of a uniform image before being analysed.

The Hough transform space is quantised: if more samples are taken in Θ, then the orientation of lines can be found more accurately, while more samples in r gives more accurate positions. However, the more samples there are in the Hough transform, the longer it will take to compute: for an image that is NX by NY pixels, with $N\Theta$ samples in Θ, and NR samples for r, the number of operations required is proportional to $NX.NY.N\Theta.NR$. For initial processing of an image, coarse sampling in r and Θ is acceptable in order to find approximate lines. Finer sampling can then be used if more accurate values for the parameters of particular lines are required.

Figure 3.1 shows an image consisting of a single straight line and Fig 3.2 its normalised Hough transform. The peak in the transform corresponding to a line having r=20, and $\Theta=\pi/6$ is clearly visible.

Having obtained the Hough transform, we then have to "invert" it to obtain the lines in the image. However, it is not possible to do this directly, since the location of a peak in the Hough transform gives the orientation and position of the line, but contains no information about its end points. To overcome this problem, the following algorithm was developed. For a given line, all the pixels lying on this line in the image were extracted into a single vector. Then, for all possible start and end points, subject to the line being a minimum length, in this vector, the mean intensity was calculated. The positions of the start and end points corresponding to the maximum mean were recorded, and used as the start and end points for this line segment. When a pixel was assigned to a line, its contribution to the Hough transform was removed, so that further peaks in the transform, corresponding to different lines, could be located.

4 DYNAMIC PROGRAMMING

Section 2 showed that a successful line finding algorithm must be capable of spanning gaps of approximately three pixels. To do this, it must take a "global" view of the image. One way in which this could be done in principle is to generate all possible line segments of length N pixels in the image, and to calculate a figure of merit for each segment. The figure of merit could be the summed intensity along the line, with a factor that penalises highly curved lines. However, this approach is not feasible, since even for an image of only 64 by 64 pixels, there are clearly far too many possible line segments of length 10 pixels, say, for the figure of merit to be calculated for all of them. The method of dynamic programming (eg Montanari 1971), overcomes this combinatorial explosion.

4.1 IMPLEMENTATION

Suppose the figure of merit for a line segment of length N is the summed intensity over the line, with a penalty for highly curved lines. Then the desired line segment is given by the values of $z_1, z_2, \ldots\ldots z_N$ and $d_1, d_2, \ldots\ldots d_N$ which maximise

$$\sum_{j=1}^{N} I(z_j) - q \sum_{j=2}^{N} (d_j - d_{j-1}) \bmod 8 \qquad (4.1)$$

where $z_j=(x_j,y_j)$ is the coordinate in the image
$I(z_j)$ is the intensity at pixel (x_j, y_j),
q is the penalty factor for curves in the line,
d_j is the exit direction from the pixel z_j, which lies in the range 0 to 7, defining the directions as follows

```
      0
    7   1
  6   zj  2
    5   3
      4
```

A pair of variables z_j, d_j are referred to as a state.

There are two constraints that must be satisfied by the variables z_j and d_j:

$\max(|x_j - x_{j-1}|, |y_j - y_{j-1}|) = 1$, which ensures that pixels on a line are contiguous.

$((d_j - d_{j-1}) \bmod 8) \leq 1$, which prevents the line from turning through a right angle in one pixel.

The calculation proceeds over N stages as follows. At stage j, we have to decide whether a state z_j, d_j should be associated with the j^{th} point of the curve. We make this decision on the basis of the profit of associating this state with the $(j-1)^{th}$ point of the curve. Thus at each stage we compute

$$V_j(z_i, d_k) = \max(I(z_i) - q\,(d_k - d_l) \bmod 8) + V_{j-1}(z_h, d_l),$$

where d_k is the exit direction from z_i,
d_l is the entry direction into z_i, and the exit direction from z_h,
z_h is the neighbour of z_i along d_l.

The calculation starts with $V_0(z_i,.) = I(z_i)$. At each stage, a memory is updated which records the path to a given state. This is used to trace back along the line.

Having completed N stages, we find the largest profit value, which gives the end point of the best line in the image:

$$P_N = \bar{x}_i, \bar{y}_i \text{ s.t. } V_N(\bar{z}_i.,) = \max_{z_i} V_N(z_i.,).$$

Having found the end point of this line, we can then trace back through the states that brought us to this point, using the memory described above. At each stage in the traceback, when a pixel is included in a line, its effect on the profit memory, V, is removed, so that it cannot be included in another line.

This algorithm is expressed in a stylised language in appendix 2. It will be clear that the method makes rather large demands on memory to record the traceback information, and also requires a large number of operations. To find lines of length N in an image of MX by MY pixels requires N.MX.MY.8.3 traverses of the inner loop. The factors 8 and 3 arise from the fact that there are 8 possible exit directions from a pixel, and for each exit direction there are 3 possible entry directions. The inner loop requires one addition and one subtraction of image values, which can be stored as fixed point numbers. Bertolazzi and Pirozzi (1984) describe how the algorithm can be implemented on a parallel architecture: if there are sufficient processors such that one pixel can be associated with one processor, then the algorithm runs in time O(N). If there are P processors, the run time is O(N.MX.MY/P), ie the algorithm increases in speed proportional to the number of processors available. The memory requirement of the algorithm is mainly determined by the traceback memory. Although the profit values must be recorded, they do not need to be stored for every stage, but can be double-buffered so that the profit at the current stage can be evaluated from the previous stage. This double buffer uses 2.MX.MY.8 storage locations. For image values in the range 0-255 (ie 8 bit data), and for N=16, for example, each storage location will require 12 bits. The memory required for traceback needs to be MX.MY.8.N locations, but each location need be only three bits, since it holds a direction in the range 0-7.

5 RESULTS

This section presents the results obtained when the algorithms described in sections 3 and 4 were applied to synthetic and real image data. For both types of data, a threshold was applied to the images as follows: the mean value of the image was calculated, and only points that exceeded this mean value were used in calculations. This has the effect of reducing the run time of the programs.

5.1 SYNTHETIC DATA

The test image used in this case consisted of a spiral imposed on a background. The intensity of the spiral was 10 times that of the background. This form of test image was chosen as it contains line segments of differing

curvatures. Fig 5.1 shows the test image, displayed in modulus form to reduce the dynamic range of the image. A thresholded version of the image is shown in Fig 5.2 with the threshold set according to the results in section 2. This shows the dropouts that occur in the line.

5.1.1 HOUGH TRANSFORM APPLIED TO SYNTHETIC DATA

The Hough transform of the image is shown in Fig 5.3. The stripes in the transform correspond to lines in the image whose distance from the origin increases as the angle increases, as would be expected for a spiral. The reconstructed image is shown in Fig 5.4. All points in the Hough transform having a value greater than the mean of the original image were used to generate the lines shown in the reconstructed image. This corresponds to reconstructing lines whose mean intensity is greater than the image mean value.

The line segments towards the edge of the spiral have been located fairly successfully, while the more central ones have not. This is to be expected, since the Hough transform is matched to straight lines, and it is only at the edge of the spiral that the line segments appear reasonably straight. This is a major problem with the Hough transform as implemented here; it can only locate straight line segments. It is possible to modify the Hough transform to locate any parameterisable shape, but it cannot find a line of arbitrary, and possibly variable curvature.

5.1.2 DYNAMIC PROGRAMMING APPLIED TO SYNTHETIC DATA

We now examine the results of applying the dynamic programming algorithm to the synthetic image. As described in section 4, this method is capable of locating curved lines. Fig 5.5 shows the lines located in the synthetic image when no curvature penalty was applied (ie q=0). Although some line segments have been located correctly, there are many loops where the algorithm has joined a pixel to itself via the shortest possible route. To eliminate these loops, the curvature penalty was applied, with q=50 (the image values are scaled in the range 0-255). The reconstructed image obtained in this case is shown in Fig 5.6. The loops have beem eliminated using the curvature penalty, although it must be remembered that the curvature penalty will prevent the algorithm locating highly curved lines.

5.2 SAR IMAGE DATA

5.2.1 PREPROCESSING OF RSRE SAR IMAGES

Fig 5.7 shows a section from a SAR image of a road at a nominal resolution of a few metres in range and azimuth. The hedges on each side of the road are clearly visible. It will be noticed that there are correlations in the vertical direction in the image, corresponding to lines of constant azimuth position. These are due to the fact that the range dimension sampling is about two thirds of the range resolution. In order to remove these range correlations, the frequency spectrum of the image in the range direction was whitened, as follows: a large image was selected which contained relatively little structure. From this image, vertical lines were extracted. Each line was Fourier transformed, and the resulting spectra were summed in power. The spectrum of the image in Fig 5.7 was then obtained, and each line of constant azimuth frequency was divided by the root mean square power spectrum obtained from the large image. This modified Fourier transform was then inverted to give the decorrelated image. The result

of this process is shown in Fig 5.8. It is clear that the range correlations have been significantly reduced while preserving the structure of the image. The line finding algorithms were then applied to this image.

5.2.2 HOUGH TRANSFORM APPLIED TO SAR IMAGE DATA

The Hough transform of Fig 5.8 is shown in Fig 5.9. The peaks in the transform due to the hedges on each side of the road are clearly visible. The reconstructed image from this transform is shown in Fig 5.10. Although portions of the hedges have been located, the majority have been missed. This is because the road is curved, resulting in broadened, lower peaks in the Hough transform space. This again demonstrates the "tuned" nature of the Hough transform.

5.2.3 DYNAMIC PROGRAMMING APPLIED TO SAR IMAGE DATA

The lines in the image of Fig 5.8 reconstructed by the dynamic programming technique are shown in Figs 5.11 and 5.12 corresponding to 10 and 20 stages respectively. For the 10 stage image, although portions of the hedges have been located correctly, there are many spurious lines located in the background. These "clutter" lines have been largely eliminated in the 20 stage image. This effect is due to another property of SAR images: the speckle noise in the background results in occasional bright pixels, leading to an increased mean brightness over short lines passing through these pixels. Thus the line finding algorithm locates short lines in clutter. This can be ovecome by locating longer line segments, but then true short line segments can be missed.

The problem of lines being located in clutter also raises the question of when the traceback process should be terminated. The traceback is started from the pixel having the highest current value in the profit array, V. Apart from the curve penalty factor, q, this value divided by the number of stages is the mean value along the current line. This gives an indication of the confidence in the line, and its strength relative to the background, and hence can be used to determine whether the traceback should be done for the line. However, a single very bright point in clutter, due to an isolated strong target for example, can give rise to reasonable mean values over a line. For this reason, a further "line editing" step is probably required, in which each line segment located by the algorithm is analysed to ensure that it has reasonable properties. One simple test is to measure the variance along the line; if this is large, then the line contains a few bright pixels, and should probably be rejected. More complicated tests could also be devised based on the context of the line image. For example, if the line is thought to be a hedge, then a shadow would be expected behind it.

6 CONCLUSIONS

Two line location algorithms have been studied in this memorandum. The Hough transform was found to locate straight lines well. However, the fact that interesting lines in SAR images need not be straight limits the usefulness of the Hough transform. The dynamic programming technique locates curved and straight line segments well, but is expensive in terms of storage and cpu time required. A multiprocessor implementation of the technique can reduce the run time significantly.

Apart from the problems of gaps in lines, speckle can also give rise to spurious lines in the background. These lines can be suppressed by demanding that the lines located have a minimum length. For the lines segments studied here, which had a target to background intensity ratio of approximately 10 to 1, it was found that a line had to be approximately 20 pixels long before it could be located. Further work could investigate an "editing" process to be applied to the lines to eliminate these spurious lines.

7 REFERENCES

Bertolazzi P and Pirozzi M 'A Parallel Algorithm for the Optimal Detection of a Noisy Curve' Comp. Vis., Graph. and Im. Proc. Vol 27 (3) 1984 pp 380-386.

Duda R D and Hart P E 'Use of the Hough transformation to Detect Lines and Curves in Pictures' Comm. ACM, Vol 15 (1) 1972 pp 11-15.

Montanari U 'On the Optimal Detection of Curves in Noisy Pictures' Comm. ACM Vol 14 (5) 1971 pp 335-345.

APPENDIX 1. PROBABILITY OF GAPS FOR LINES OF LENGTH N

GAP LENGTH	N=5	N=10	N=15	N=20	N=25	N=30	N=35	N=40	N=45	N=50
D=0.1										
1	0.407	0.644	0.798	0.878	0.928	0.956	0.978	0.984	0.991	0.994
2	0.038	0.077	0.120	0.163	0.197	0.232	0.270	0.299	0.336	0.363
3	0.004	0.007	0.011	0.017	0.022	0.027	0.031	0.035	0.041	0.044
4	0.000	0.001	0.001	0.002	0.003	0.003	0.004	0.004	0.005	0.006
5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
D=0.2										
1	0.676	0.895	0.964	0.990	0.996	0.998	1.000	1.000	1.000	1.000
2	0.133	0.274	0.396	0.489	0.568	0.643	0.699	0.748	0.789	0.822
3	0.022	0.051	0.080	0.112	0.140	0.167	0.195	0.223	0.244	0.267
4	0.003	0.008	0.014	0.020	0.024	0.031	0.037	0.042	0.047	0.052
5	0.000	0.001	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.009
D=0.3										
1	0.832	0.972	0.995	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	0.274	0.508	0.660	0.769	0.839	0.890	0.927	0.947	0.966	0.978
3	0.063	0.157	0.238	0.314	0.381	0.440	0.494	0.544	0.585	0.625
4	0.014	0.043	0.070	0.099	0.125	0.151	0.176	0.202	0.226	0.246
5	0.003	0.012	0.018	0.028	0.036	0.046	0.054	0.062	0.068	0.076
D=0.4										
1	0.921	0.995	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	0.427	0.707	0.852	0.924	0.964	0.981	0.991	0.995	0.998	0.999
3	0.140	0.314	0.453	0.561	0.648	0.720	0.775	0.821	0.858	0.883
4	0.043	0.117	0.183	0.250	0.312	0.362	0.414	0.459	0.504	0.543
5	0.011	0.042	0.075	0.100	0.127	0.156	0.183	0.205	0.231	0.258
D=0.5										
1	0.966	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	0.593	0.859	0.948	0.983	0.994	0.998	1.000	1.000	1.000	1.000
3	0.249	0.511	0.681	0.791	0.862	0.910	0.939	0.959	0.976	0.983
4	0.099	0.244	0.373	0.478	0.566	0.634	0.702	0.745	0.791	0.824
5	0.031	0.108	0.180	0.248	0.309	0.366	0.422	0.471	0.513	0.554

APPENDIX 2

This appendix presents the implementation of the dynamic programming algorithm in a stylised language.

```
for Each Stage
      for each pixel
           for each exit direction from this pixel
                for each entry direction for this exit  direction
                if [(profit from previous stage
                +    image value for this pixel
                -    curvature penalty for given entry and exit direction)
                     is a maximum] then
                          record the maximum profit
                          record the entry direction for this exit direction
                end if
                next entry direction
           next exit direction
      next pixel
next Stage
set threshold for reconstruction of lines
loop:
scan the profit array for its maximum value
if [maximum profit exceeds threshold] then
      record the pixel corresponding to the maximum profit
      obtain the direction line enters this pixel from the memory
      remove the effect of this pixel from the profit array
           for each stage starting from the last
           get the next pixel on the line from the previous pixel and its direction
           obtain the direction line enters this pixel from memory
           trace the effect of this pixel back to the final stage
           remove its effect from the profit array
           next stage
else
      stop
endif
goto loop
```

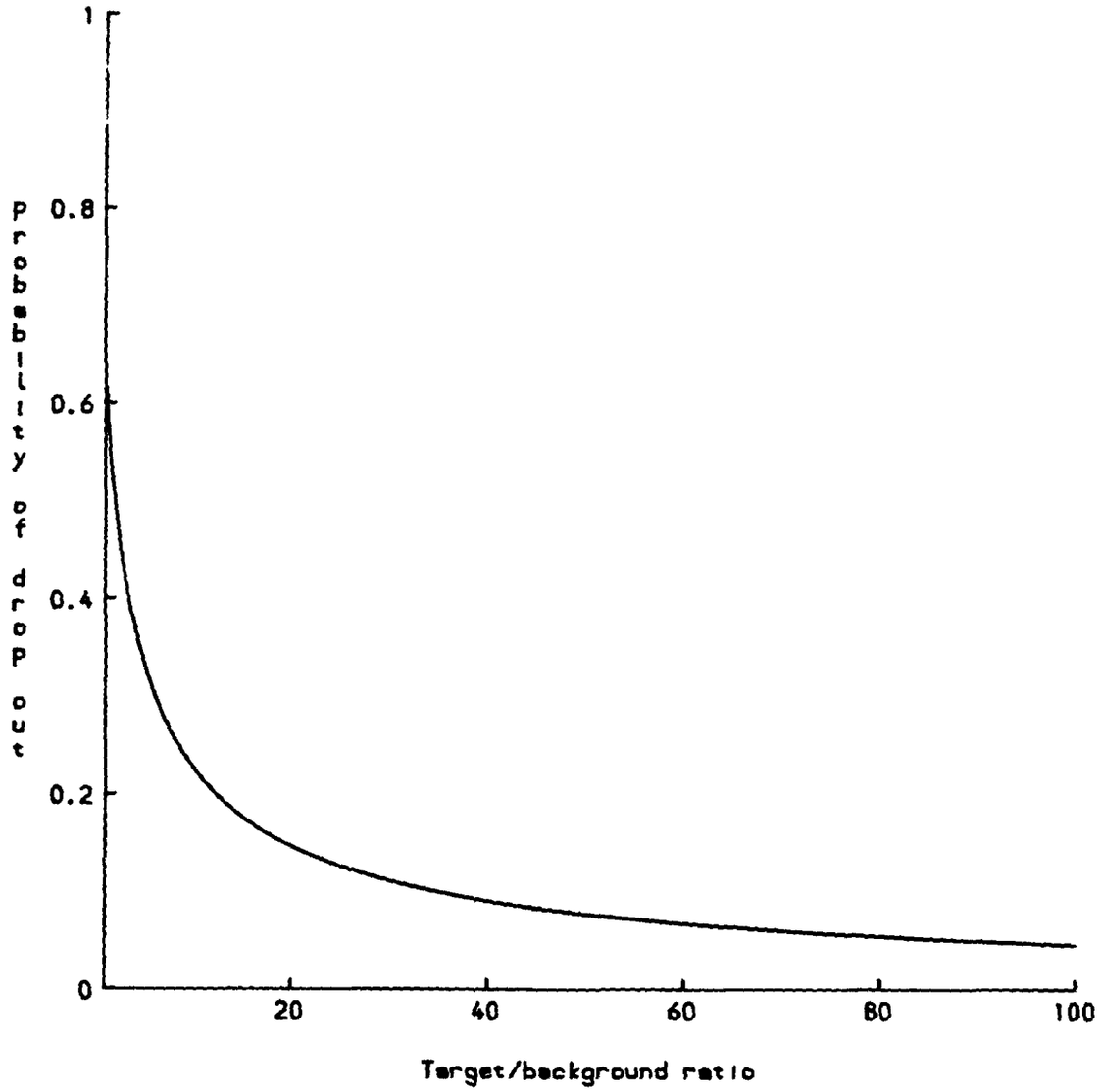


Fig. 2.1 Probability of a single pixel in a target falling below threshold

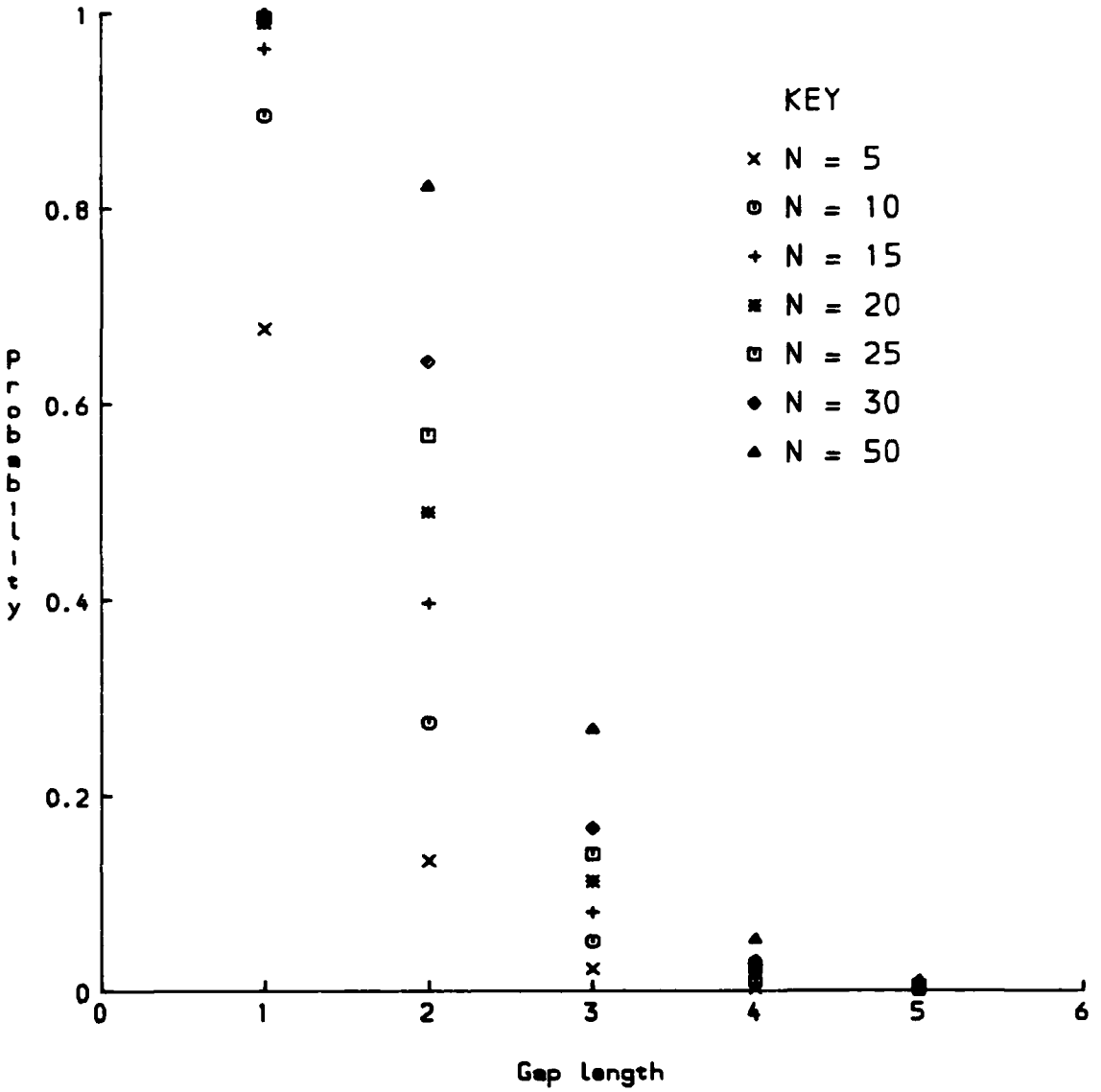


Fig. 2.2 Probability of gaps for lines of different lengths, N. Plotted for D=0.2

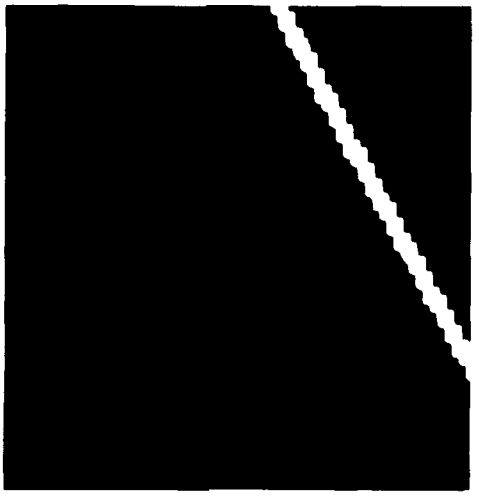


Fig. 3.1 Straight line image

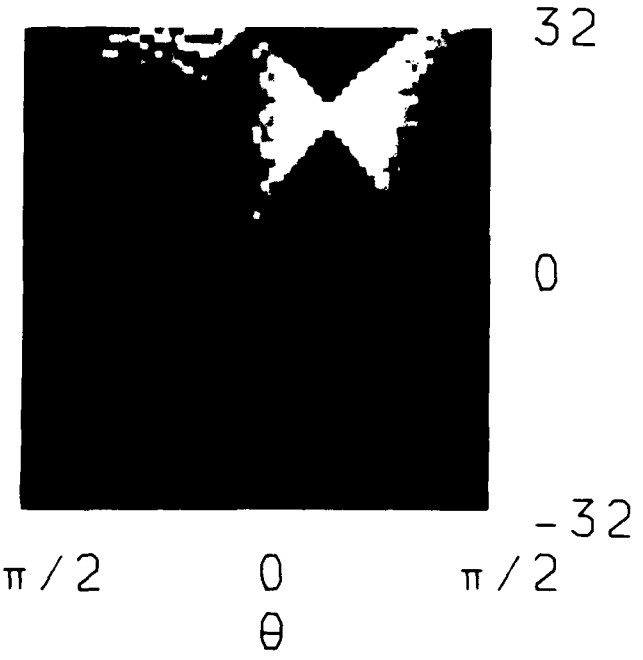


Fig. 3.2 Normalised Hough transform of Fig. 3.1

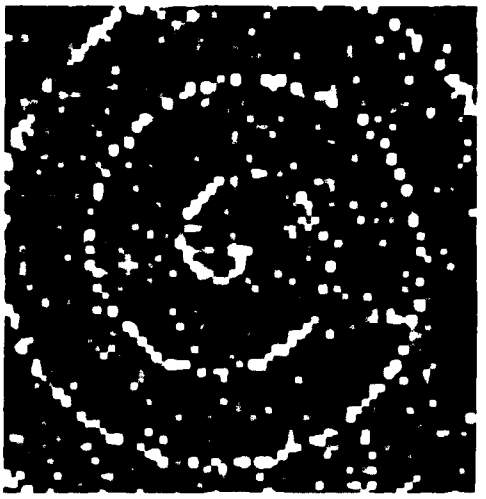


Fig. 5.1 Synthetic test image

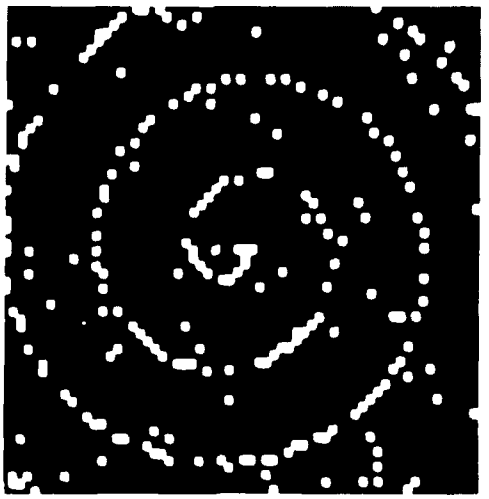


Fig. 5.2 Thresholded image

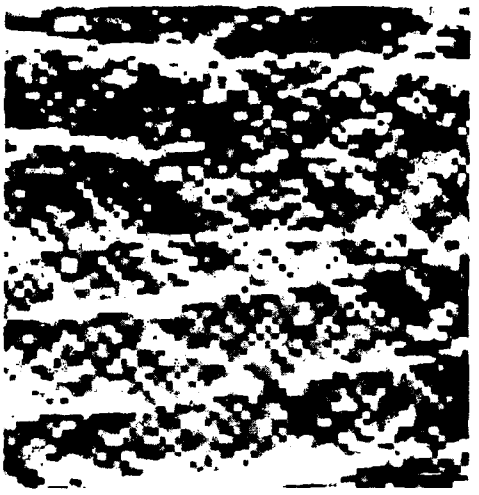


Fig. 5.3 Hough transform
of synthetic image

Fig. 5.4 Reconstructed image

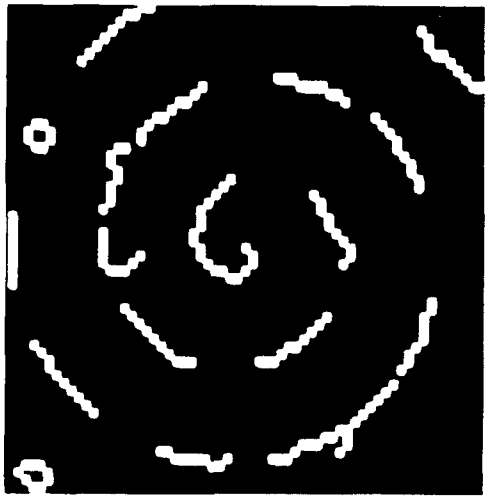


Fig. 5.5 Dynamic programming reconstruction with no curvature penalty

Fig. 5.6 Dynamic programming reconstruction with curvature penalty, q = 50

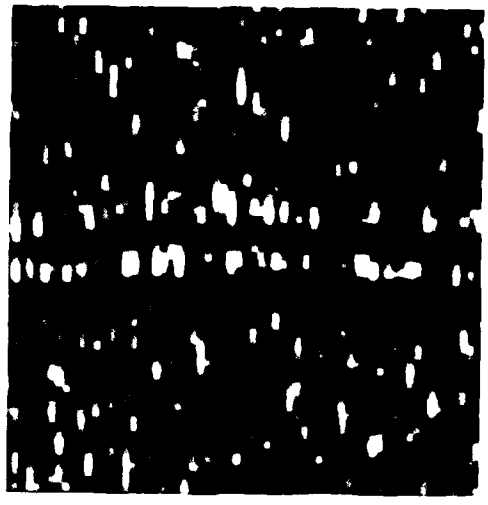


Fig. 5.7 Raw SAR image

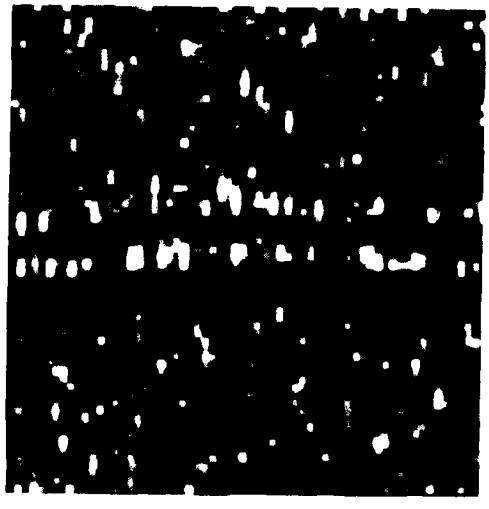


Fig. 5.8 Preprocessed SAR image

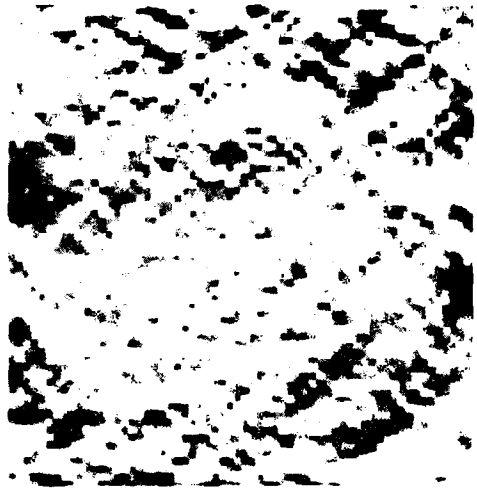


Fig. 5.9 Hough transform of preprocessed SAR image

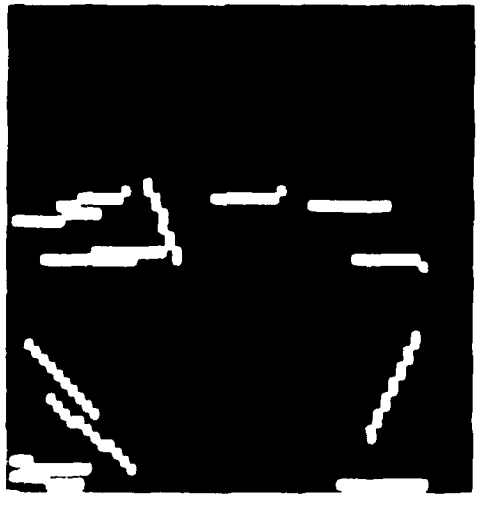


Fig. 5.10 Reconstructed image from Hough transform

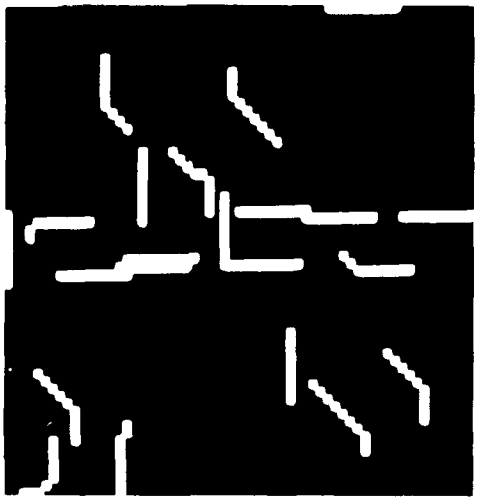


Fig. 5.11 Dynamic programming reconstruction of Fig. 5.8 for 10 stages

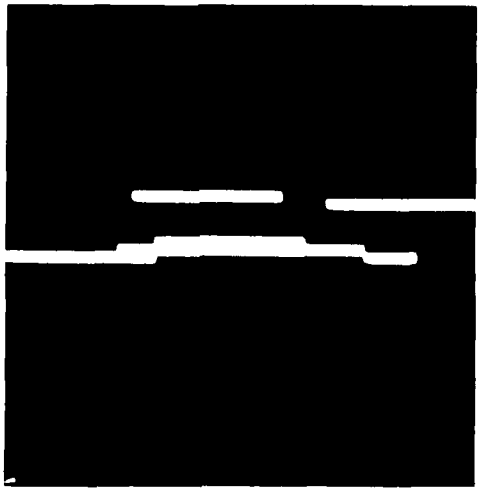


Fig. 5.12 Dynamic programming reconstruction of Fig. 5.8 for 20 stages

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference MEMORANDUM 3841	3. Agency Reference	4. Report Security U/C Classification
5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location ROYAL SIGNALS AND RADAR ESTABLISHMENT		
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location		

7. Title
LINE FINDING ALGORITHMS FOR SAR

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
WOOD J W				

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement
Hd BS Group

Descriptors (or keywords)

continue on separate piece of paper

Abstract
This memorandum addresses the problem of locating linear features, not necessarily straight, in Synthetic Aperture Radar images. The characteristics of such features and the way in which these characteristics affect the desired properties of line finding algorithms are discussed.

Two line detection algorithms are then described: the Hough transform, and the dynamic programming technique. Results of applying both algorithms to simulated and real SAR images are given. The Hough transform is found to be too restricted to be useful for this application. The dynamic programming technique locates lines well. The problem of spurious lines appearing in clutter is discussed.

END

FILMED

1-86

DTIC